CS 4476: Computer Vision, Fall 2020

PS1

Author: Saumya Jain

Date: Wednesday, September 9th, 2020

Problem 1

(1.) Give an example of how one can exploit the associative property of convolution to more efficiently filter an image.

The associative property of convolution can be represented by (f * g) * h = f * (g * h). This property is very effective to improve the efficiency of filtering an image. An example of this is with linearly separable filters. We can make the convolution of an image very efficient with two 1-D filters. This is essentially done by convolving the two 1-D filter to form a 2-D filter.

(2.) This is the input image: [0 0 1 1 0 0 1 1]. What is the result of dilation with a structuring element [1 1 1]?

Our input image is given as $\begin{bmatrix}0 & 0 & 1 & 1 & 0 & 0 & 1 & 1\end{bmatrix}$ and our structuring element is given as $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$. To get the result of dilation, we need to overlap the structuring element over our input image and perform an XOR Operation. So, the result of dilation is $\begin{bmatrix}0 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$.

(3.) The filter f' = [0 -1/2 0 1/2 0] is used as the filter to compute an estimate of the first derivative of the image in the x direction. What is the corresponding second derivative filter f". (Hint: Assymetric filters must be flipped prior to convolution.)

The second derivative filter f" is given as $\begin{bmatrix}0 & 0 & -1/4 & 0 & 1/2 & 0 & -1/4 & 0 & 0\end{bmatrix}$

(4.) Name two specific ways in which one could reduce the amount of fine, detailed edges that are detected with the Canny edge detector.

The ways to reduce the amount of fine, detailed edges detected by Canny Edge Detector are:

(i) Changing the Gaussian filter: A regular Gaussian filter has a smoothing effect on the edges and noise of an image. To reduce the fine, detailed edges, we need to use a Gaussian filter that smooths out noise more than it smooths out edges. So, we can use an adaptive Gaussian filter which calculates discontinuity in gray-scale images. This will reduce the amount of highly detailed edges being detected.

(ii) Increasing the Sigma value: By increasing the sigma value of the Canny edge detector, we can easily reduce the number of fine, detailed edges.

(5.) Describe a possible flaw in the use of additive Gaussian noise to represent image noise.

We cannot use additive Gaussian noise to represent image noise. This is because sums of Gaussian noise are not arithmetic. Instead, total noise is a quadrature sum. The overall noise is less than the arithmetic noise because there is a significant probability that noise in the future will wipe out noise from the past. If previous noise causes a signal too high, there's a high chance that adding more noise will make the signal value lower.

(6.) Design a method that takes video data from a camera perched above a conveyor belt at an automotive equipment manufacturer, and reports any flaws in the assembly of a part. Your response should be a list of concise, specific steps, and should incorporate several techniques covered in class thus far. Specify any important assumptions your method makes.

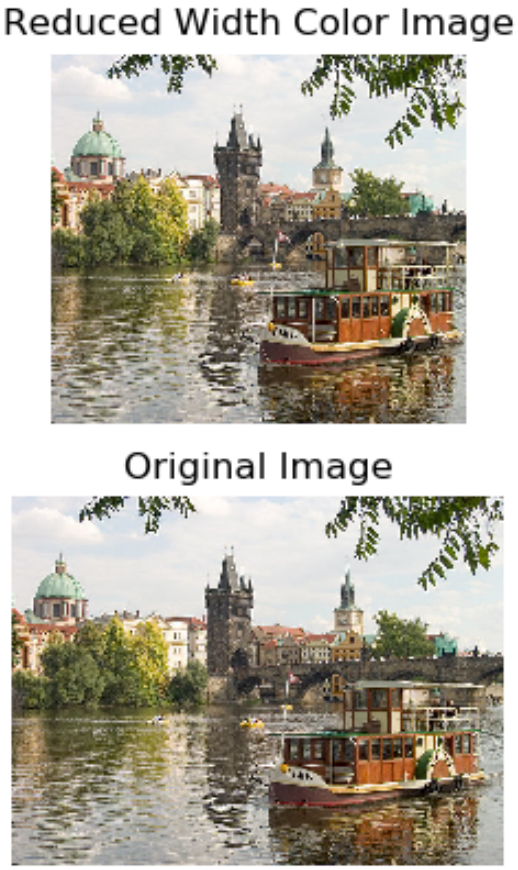
For a camera perched above a conveyor belt at an automotive equipment manufacturer, it is important to effectively spot any flaws. This will involve efficient edge detection and binary image analysis. First, the camera needs to use something like Canny Edge Detection on every part that passes from in front of it on the assembly line. A Hysteresis filter with thresholding can be used to get rid of parts with imperfections. Then, in binary image analysis, a morphological operator (like erosion or dilation) can be used to filter the threshold further. The relevant components can be taken out or extracted from the processed image and compared to preset parts in a stored database. This is assumed to have all required and relevant information on all properties of every part. Each component can be compared to the data in the database and if a match isn't found for a specific part, it will be rejected. This is an efficient way to design this system, especially when part rejection should occur rapidly before the part advances further down the assembly line.

Problem 2

Prague

Mall

The original images

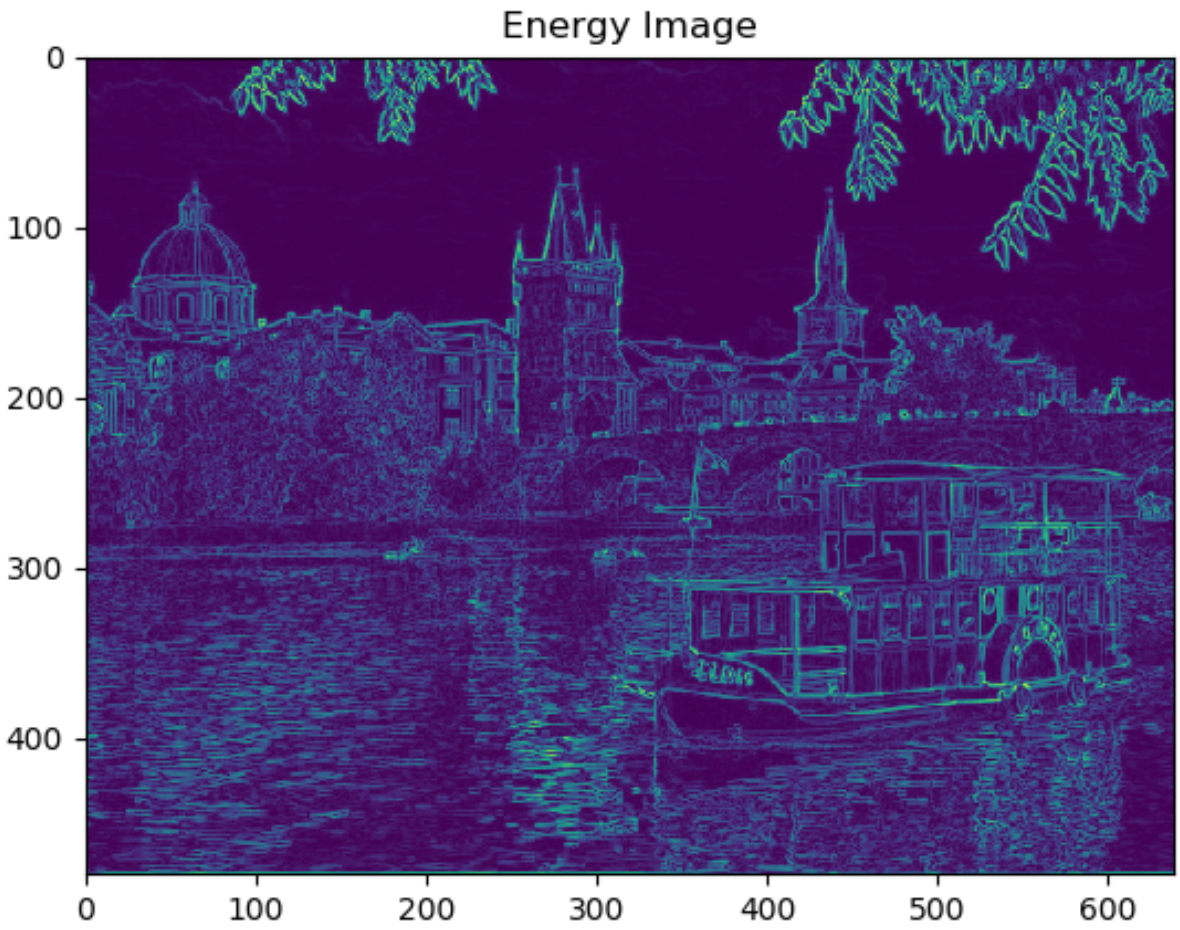


(1 a) Prague - Width 100 Pixel Reduction (seam_carving_reduce_width)

(1 b) Mall - Width 100 Pixel Reduction (seam_carving_reduce_width)

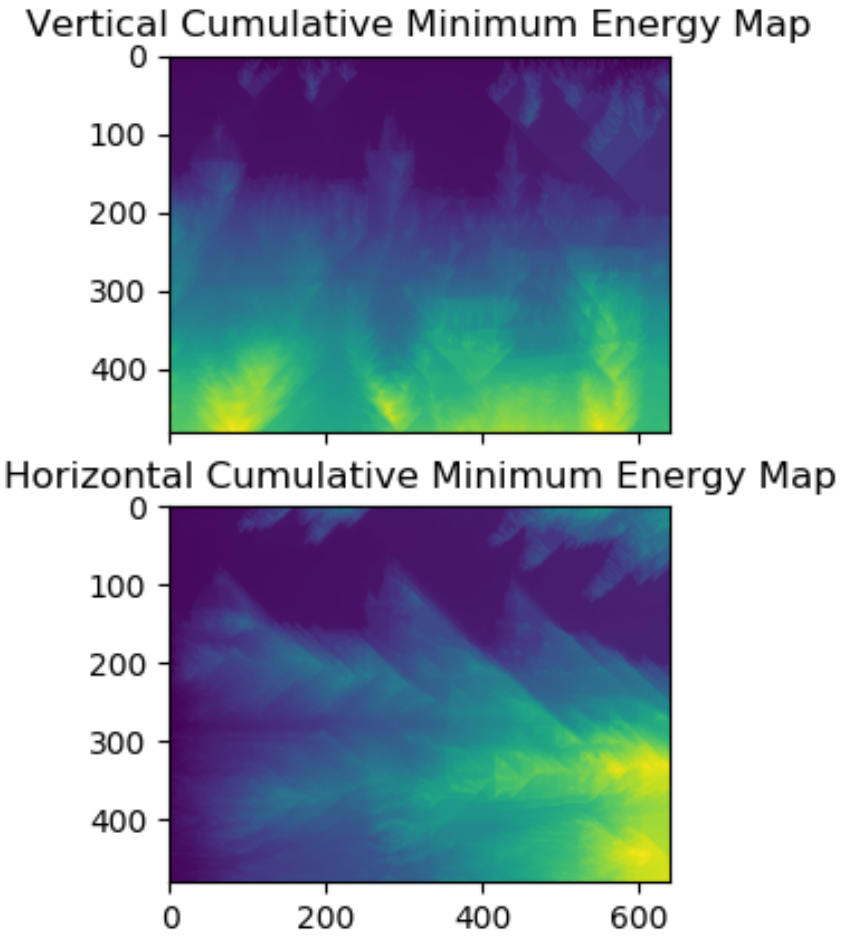
(2 a) Prague - Height 100 Pixel Reduction (seam_carving_reduce_height)

(2 b) Mall - Height 100 Pixel Reduction (seam_carving_reduce_height)



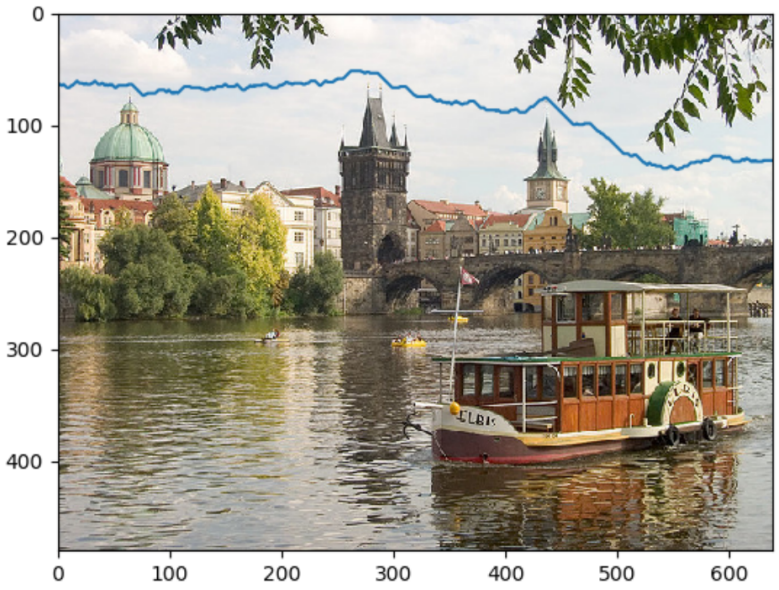
(3 a) Prague - Energy Function Output

This output displays the energy of each pixel in the image. First, the gradients in the x and y directions are calculated. This is done through calculating the approximate derivatives the input image using a differentiation filter in the x and y directions. The energy function calculates energy for each pixel by summing the absolute values of the gradients in the x and y directions. This image looks this way because the gradients are substantially high at the edges of objects and noise in the image. Hence, the resultant energies are also high. This results in the highlighting of edges and shapes by the energy function.



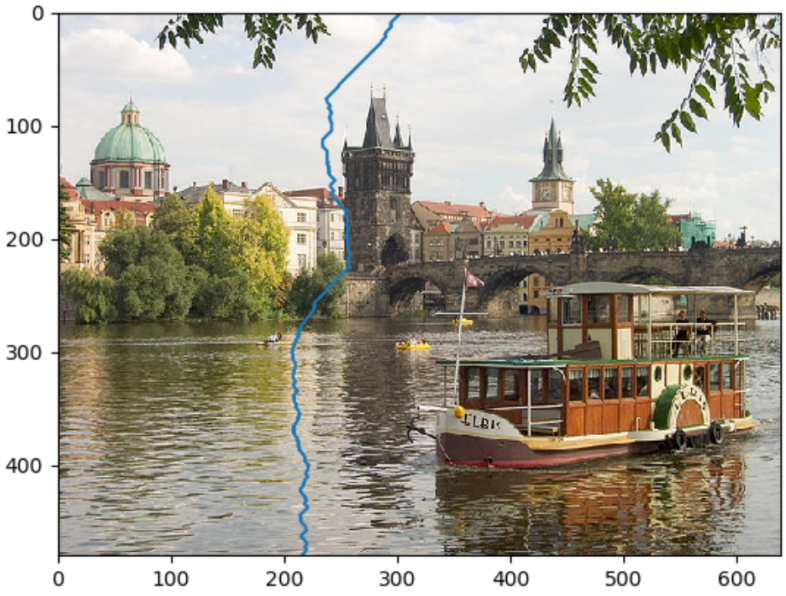
(3 b) Prague - Minimum Energy Maps

The output energy maps display a graphical representation of the sum of energies in an area of the image. Areas with a yellow/green hue have more energy than areas with a blue/black hue. This distinguishes low energy parts of the image from the high energy parts, so that we can efficiently calculate the horizontal and vertical seams.This matches the Energy Function Image from above, as parts of the image with less energy in the image subsequently have low cumulative energy in the maps.



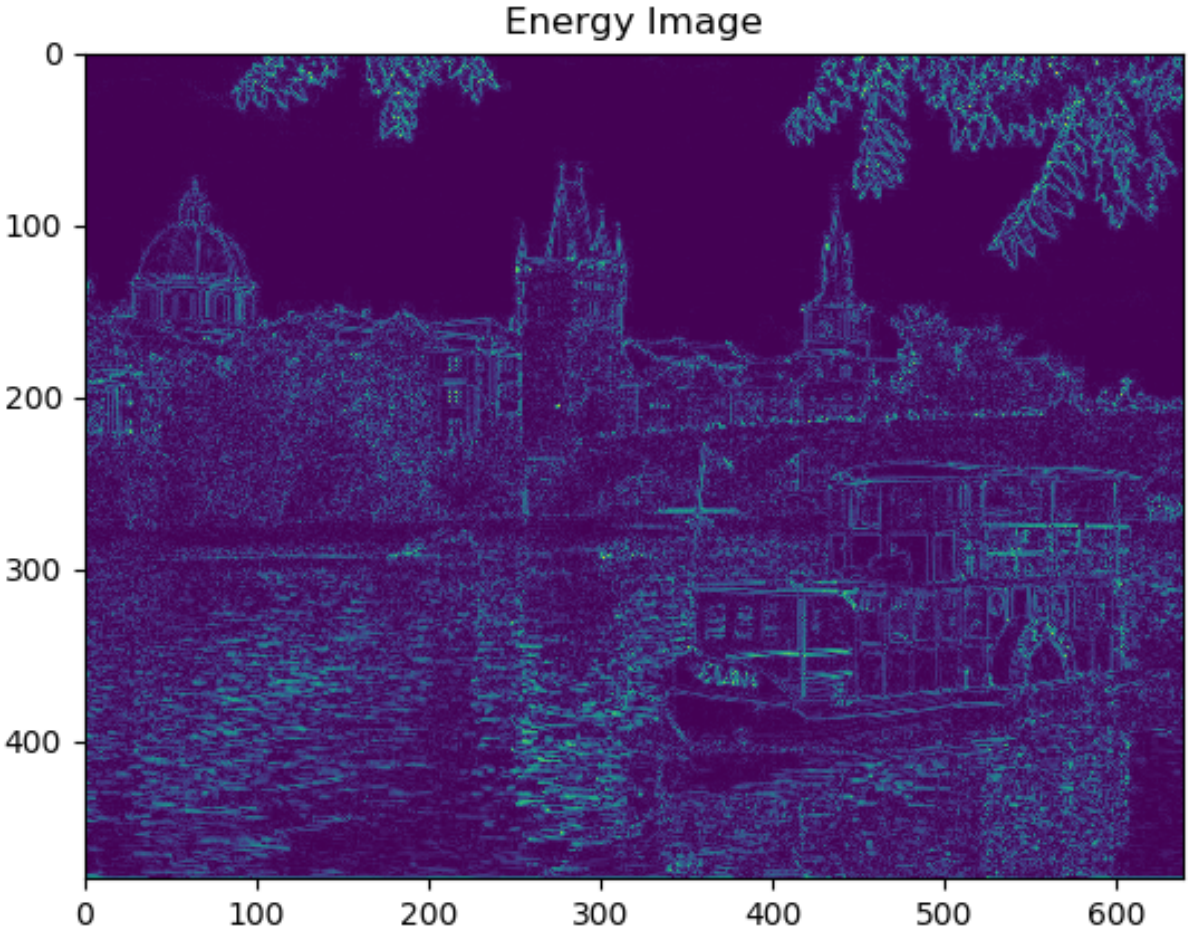
(4 A) Prague - Horizontal Seam

The horizontal seam given here is optimal as it traces a path along the areas with lowest values of cumulative minimum energy on the map from above. It chooses this path to select areas with the least cumulative energy for easy removal.



(4 B) Prague - Vertical Seam

The vertical seam given here is optimal as it traces a path along the areas with lowest values of cumulative minimum energy on the map from above. It chooses this path to select areas with the least cumulative energy for easy removal.



(5 b) Change in Energy Function - Laplacian Filter

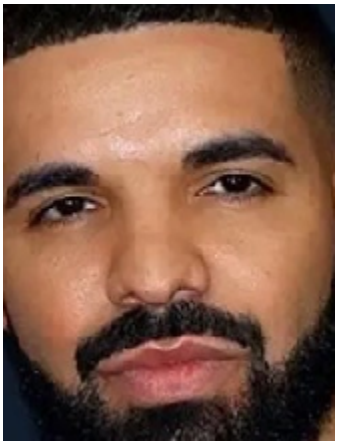
(a) Instead of using the Prewitt filter to compute gradients, I changed the code to instead use a Laplacian filter. This ends up changing the gradients and hence changing the energy image.

(b) The resulting energy image can be seen above.

(c) We can see a difference in the energy image produced by the Laplacian Filter. The edges are no longer as clearly defined. Earlier, we used a Prewitt filter which measures the slope as it is a 1st order derivative operator. Now, the Laplacian filter measures the change of the slope as it is a 2nd order derivative operator. The Laplacian filter is more sensitive to noise as a result.

(6 a) Original Input Image - Bad Example - Drake Face

(6 b) Resized Image (SeamCarving) - Bad Example - Drake Face

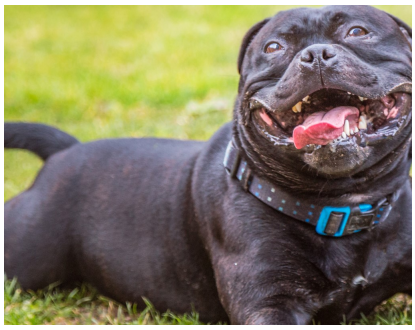


(6 c) Resized Image (Simple Resampling) - Bad Example - Drake Face

(6 d) Original Image Dimensions: 265 x 350 (W x H)
Resized (SeamCarving) Image Dimensions: 165 x 250 (W x H)

(6 e) Sequence of Removals: Reduced width 100 times (-100 pixels) and then reduced height 100 times (- 100 pixels)

(6 f) Through Seam Carving in the output, we see that the free skin in Drake's face has disappeared due to low energy. We are left with only his main facial features like mouth, eyes and nose. This is a bad example of Seam Carving.



(6 a) Original Input Image - Good Example - Dog

(6 b) Resized Image (SeamCarving) - Good Example - Dog

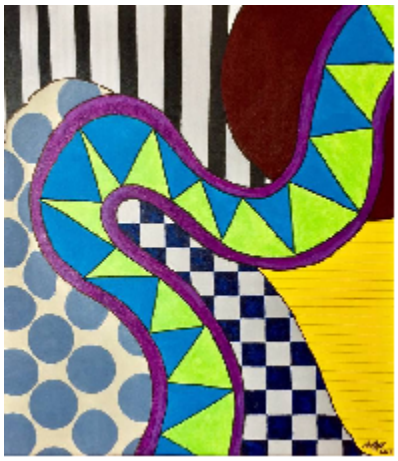
(6 c) Resized Image (Simple Resampling) - Good Example - Dog

(6 d) Original Image Dimensions: 1019 x 796 (W x H)
Resized (SeamCarving) Image Dimensions: 919 x 696 (W x H)

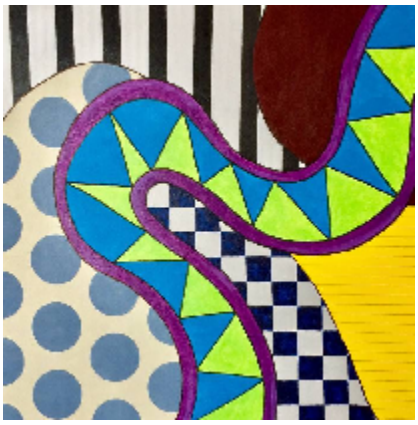
(6 e) Sequence of Removals: Reduced width 100 times (-100 pixels) and then reduced height 100 times (- 100 pixels)

(6 f) Through Seam Carving in the output, we see that some of the dog's belly has reduced in size. However, this has not had any negative effect on the image. We now have a smaller cropped image without losing any details of the original image. If we had instead used simple resampling, we can see the significant loss of details.

(6 a) Original Input Image - Bad Example - Pattern



(6 b) Resized Image (SeamCarving) - Bad Example - Pattern



(6 c) Resized Image (Simple Resampling) - Bad Example - Pattern

(6 d) Original Image Dimensions: 780 x 770 (W x H)
Resized (SeamCarving) Image Dimensions: 680 x 760 (W x H)

(6 e) Sequence of Removals: Reduced width 100 times (-100 pixels) and then reduced height 10 times (- 10 pixels)

(6 f) Through Seam Carving in the output, we see that the circular pattern in the left side of the image has decreased in distance from one another. Since this is a painting, this is an undesirable effect as it is essentially changing the painting. In this case, a simple resampling is better through cropping.

(6 a) Original Input Image - Good Example - Mountains

(6 b) Resized Image (SeamCarving) - Good Example - Mountains

(6 c) Resized Image (Simple Resampling) - Good Example - Mountains

(6 d) Original Image Dimensions: 590 x 394 (W x H)
Resized (SeamCarving) Image Dimensions: 490 x 294 (W x H)

(6 e) Sequence of Removals: Reduced width 100 times (-100 pixels) and then reduced height 100 times (- 100 pixels)

(6 f) Through Seam Carving in the output, we see that the initial picture of the mountains has become smaller without losing its significant details (main mountain range, clouds in the sky, trees, snow). We have only taken out the less significant areas of the picture.